

## الفصل التاسع استخدام XML داخل التطبيقات

لعلك قرأت في الفصول السابقة عن لغة XML وكيف أن نطاق .NET يعتمد عليها اعتماداً كبيراً. لذا رأينا أن نقوم بإطلاله سريعة على هذه اللغة أو هذا التنسيق حتى تتمكن من التعامل معه أثناء تطوير تطبيقاتك. وللعلم عزيزي القارئ فإن هذه اللغة تحتاج على الأقل إلى مرجع كامل لشرحها وتوضيح خصائصها. بانتهاء هذا الفصل ستتعرف على:

- مفهوم XML.
- ميزات استخدام XML.
- مسميات XML.
- خصائص XML.
- كتابة ملفات XML باستخدام التصنيف `XmlTextWriter`.
- قراءة ملفات XML باستخدام التصنيف `XmlTextReader`.

خطت لغة الترميز الممتدة (XML) Extensible Markup Language خطوات سريعة إلى أن أصبحت أحد السمات القياسية في عالم الحاسبات. سنقوم في هذا الفصل بإلقاء الضوء على لغة XML وكيفية استخدام نطاق .NET لإنشاء ومعالجة بيانات XML.

للتعرف على المزيد عن لغات الترميز ومن بينها لغة الترميز الممتدة XML، راجع أحد كتبنا في هذا المجال مثل كتاب "برمجة الإنترنت باستخدام XHTML".



## مفهوم XML

تعتبر XML إحدى الطرق المستخدمة في تمثيل البيانات المركبة. فهي تشابه مع لغة HTML في استخدامها للعناصر والرموز؛ أو ما نطلق عليه Tags، لتمثيل كل جزء من أجزاء البيانات. ولكن على عكس HTML، يمكنك داخل XML تعريف عناصر جديدة بالأسماء التي تروق لك والتي تتناسب مع بياناتك. ولعل أحد الميزات الهامة داخل XML هو تعبير بيانات ملفات XML عن نفسها وذلك لتضمن كل عنصر بيانات داخل رمز مستقل. للتعرف على الشكل العام لكود XML، دعنا نرى الكود التالي الذي يحتوي على بيانات عن السيارات:

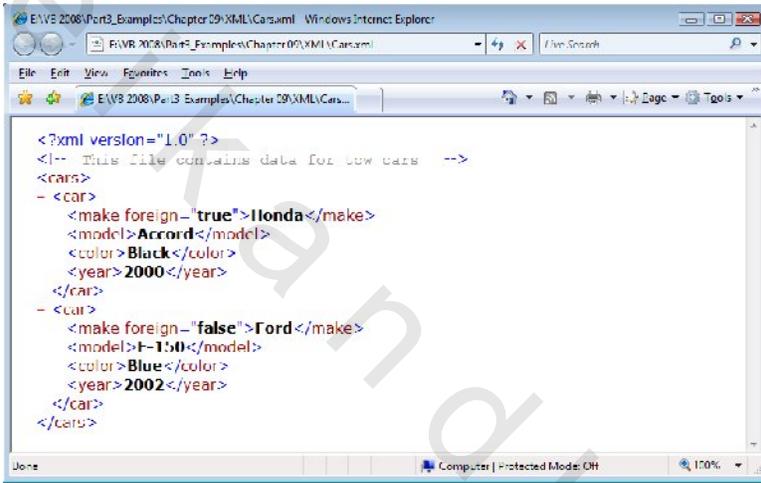
```

1. <?xml version="1.0" ?>
2. <!-- This file contains data for tow cars -->
3. <cars>
4.     <car>
5.         <make foreign="true">Honda</make>
6.         <model>Accord</model>
7.         <color>Black</color>
8.         <year>2008</year>
9.     </car>
10.    <car>
11.        <make foreign="false">Ford</make>
12.        <model>F-150</model>
13.        <color>Blue</color>
14.        <year>2010</year>
15.    </car>
16. </cars>
    
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ١ تم تعريف XML حيث يجب أن يحتوي أي ملف XML على هذا التعريف في بداية الملف. وحتى كتابة هذه السطور، يستخدم الجميع الإصدار الأول من XML وسيبقى ذلك لبعض الوقت إلى أن يأتي إشعاراً آخر.
- في السطر رقم ٢ تم وضع تعليق داخل ملف XML يحتوي على وصف لمحتويات الملف، حيث يمكنك إنشاء التعليقات في أي مكان داخل الملف باستخدام نفس الصيغة المستخدمة في HTML.
- يحتوي السطران ٣ و ١٦ على الرمز `<cars>` و `</cars>` اللذين يكونان عنصر XML. وقد يحتوي عنصر XML بداخله على عنصر XML أو أكثر. ولأن العنصر `<cars>` غير موجود داخل أي عنصر آخر، فهو يسمى العنصر الرئيسي `Root Element`. ويحتوي العنصر `<cars>` في هذا المثال على عنصرين `<car>` والتي يطلق عليها العناصر الوليدة `Children Elements` والتي تنتمي بدورها إلى العنصر الأبوي `<cars>`. كما يحتوي كل عنصر `<car>` بدوره على مجموعة من العناصر الوليدة مثل `<make>` و `<model>`. وحقيقةً فإن XML لا تضع أي قيود على إنشاء العناصر الوليدة، فمن الممكن أن يحتوي العنصر على أي عدد من العناصر الوليدة.
- يحتوي السطر رقم ٥ على العنصر `<make>` الموجود داخل العنصر `<car>` والذي لا يحتوي على عناصر أخرى في هذه المرة وإنما يحتوي على بيانات نصية (كلمة Honda) لذا يطلق عليه عنصر نصي `Text Element`، كما يحتوي هذا العنصر أيضاً على صفة `Attribute` تسمى `foreign` والتي تم تخصيصها بالقيمة `true`. وعامةً من الممكن أن يحتوي العنصر على أي عدد من الصفات.

- لاحظ احتواء جميع العناصر على رمزين، أحدهما لبداية العنصر والآخر لنهايتيه وهذا الأمر لم يكن ضرورياً في HTML إلا أنه من الأشياء الأساسية داخل XML إلا في حالة استخدام الصيغة المختصرة كما سنرى بعد قليل.
- يمكنك عرض ملفات XML داخل نافذة مستعرض الويب، وهذه الخاصية متاحة بدءاً من Internet Explorer 5.0. يمكنك عرض الملف بكتابة عنوانه مباشرة داخل شريط عنوان المستعرض كما في شكل ٩-١.



شكل ٩-١ عرض ملف XML داخل نافذة المستعرض

## مميزات استخدام XML

- ترجع شهرة XML واستخدامه دوناً عن لغات الترميز والتقنيات الأخرى للعديد من السمات التي يتمتع بها والتي من أهمها ما يلي:
- يبنى XML أساساً على النصوص، لذا فإن عمليات تصحيح الكود الذي يقوم بإنشاء XML أو استخدامه تتم بسهولة متناهية.
- من السهل نقل بيانات XML من حاسب إلى آخر تماماً مثل HTML.
- يبنى XML على القياسات الصناعية المحددة من قبل مجموعة اتحاد ناشري الويب W3C وهو ما يعني إمكانية تشغيل واستخدام XML مع جميع الأجهزة والبرامج.



تتكون مجموعة اتحاد ناشري الويب من عدد من الخبراء وتعمل على إنشاء الأشياء القياسية (التوصيات) التي يجب الحفاظ عليها عند العمل مع الإنترنت مثل HTML و XHTML و XML. يمكنك التعرف على هذه المجموعة عن قرب من خلال زيارة العنوان التالي:

<http://www.w3c.org>

- يمكنك استخدام XML في أى مكان داخل برامج .NET. والتي من أهمها الأماكن الآتية:
- يمكنك تحويل أى كائن .NET. إلى التمثيل المكافئ باستخدام XML والعكس صحيح.
- يتم تخزين بيانات التهيئة Configuration Data الخاصة بجميع برامج .NET. بما في ذلك ASP.NET وخدمات الويب بتنسيق XML.
- يتم استدعاء وظائف خدمات الويب باستخدام البروتوكول SOAP الذى يستخدم بدوره XML.
- يستخدم XML حتى تتمكن برامج العميل من استكشاف إمكانيات خدمة الويب.
- تستخدم ADO.NET لغة XML ومخططات XML للتحقق من صحة البيانات.
- يمكنك استخدام مخططات XML لإنشاء التصنيفات.

## مسميات XML

يعتبر كل رمز من الرموز التي تستخدمها داخل ملفات XML جزء من قاموس XML الذى تقوم بتعريفه بنفسك. ففي الكود السابق على سبيل المثال، يتكون قاموس XML من المصطلحات <cars> و<car> و<make> و<model> و<color> و<year> و<foreign>. ولأن كل منا محول في إنشاء القاموس الذى يحتوى على مصطلحات XML التى يقوم باستخدامها، فمن الممكن أن يستخدم كل مطور أو مؤسسة نفس رمز XML لوصف نوع مختلف من البيانات، وبالتالي تكون هناك فرصة للتعارض إذا ما وجد رمزين بنفس الاسم ويستخدمان في أغراض مختلفة. ولحل هذه المشكلة، يمكنك تضمين مسمى Namespace مع كل ملف XML، حيث يتم تعريف هذا المسمى من

قبل "معرف الموارد المنتظم" (URI) **Uniform Resource Identifier** والذي يمثل عامةً عنواناً على الإنترنت مثل <http://www.w3.org> كما يمكن أن يحتوي على محتويات أحد أدلة الإنترنت مثل <http://www.w3.org/TR/xmlbase> كما يمكن أن يحتوي على عنوان بريد إلكتروني أيضاً.

يمكنك مصاحبة (تضمين) مسمى داخل ملف XML من خلال الصفة **xmlns** وذلك داخل أول رمز ترغب في تطبيق هذا المسمى عليه. يوضح الكود التالي كيفية تضمين مسمى داخل الكود السابق:

```
<?xml version="1.0" ?>
<cars xmlns="http://www.compuscience.com/carproject">
  <car>
    <make foreign="true">Honda</make>
    <model>Accord</model>
    <color>Black</color>
    <year>2009</year>
  </car>
</cars>
```

ففي هذا الكود يتم تضمين جميع العناصر الموجودة داخل العنصر **<cars>** بالمسمى **carproject** الموجود داخل العنوان <http://www.compuscience.com>. يمكنك أيضاً تضمين المسمى مع عناصر معينة. لتوضيح ذلك، دعنا نرى الكود التالي الذي يتم فيه تضمين مسمى مع العناصر **<make>** و **<model>** فقط.

```
<?xml version="1.0" ?>
<cars xmlns:S="http://www.compuscience.com/carproject">
  <car>
    <S:make foreign="true">Honda</S:make>
    <S:model>Accord</S:model>
    <color>Black</color>
    <year>2009</year>
  </car>
</cars>
```

وقد قمنا باستخدام الحرف S لتعريف الرموز التي تستخدم المسمى. ويمكنك بالطبع استخدام أى حرف آخر أو حتى كلمة كاملة.

## خطأ XML

هناك بعض الأشياء الأساسية داخل XML والتي يجب أن تعرفها قبل الدخول في تصنيفات XML الموجودة داخل نطاق .NET. ، حيث يجب أن تعرف جميع وثائق XML تعريفياً صحيحاً وهو ما يعنى التزامها بالقواعد التالية:

- من طبيعة رموز XML أنها حساسة لحالة الأحرف، وهذا يعنى اختلاف الرمز <Cars> عن الرمز <cars> نظراً لاختلاف حالة الحرف C في كلا الرمزين.
- يجب أن يصاحب كل رمز مفتوح برمز آخر لإغلاقه والعكس صحيح. فالكود التالي على سبيل المثال غير صحيح:

```
<car>  
  <name>BMW  
</car>
```

وذلك لعدم وجود رمز إغلاق الرمز المفتوح <name>.

ولكن يستثنى من هذه القاعدة العنصر الخالى أو العنصر الذى يحتوى على صفات فقط، ففي هذه الحالة يحتوى الرمز على قوس الإغلاق ">" مسبقاً بالعلامة "/" كما فى الكود التالى:

```
<car domestic="true" color="red" />  
<car domestic="false" color="blue" />
```

- يحتوى XML على العديد من الحروف المحجوزة بما فى ذلك الحروف "<" و ">" و "&" وكلمة XML نفسها. كما يجب ألا يبدأ أى رمز بالحروف XML كما يجدد كل رمز بالحروف "<" و ">" كما فى الكود السابق.

يمكنك استخدام الرمز CDATA إذا أردت تضمين الحروف المحجوزة داخل بيانات XML كما فى الكود التالى:

```
<description><![CDATA[ This tag contains data with  
reserved characters. They are : < > & ]]></description>
```

لاحظ أن الرمز **CDATA** يبدأ بالتركيب **[CDATA[!<]** وينتهي بالتركيب **>]]**. كما يمكنك استخدام بعض التراكيب الأخرى للحصول على هذه الحروف مثل الترييب **&lt;** للحصول على الحرف "<" والترييب **&gt;** للحصول على الحرف ">" والترييب **&amp;** للحصول على الحرف "&". وبذلك يمكنك إعادة كتابة الكود السابق كما يلي:

```
<description>This tag contains data with reserved characters. They are: &lt; &gt; &amp;</description>
```

- يجب وضع عناصر XML في شكل هرمي يحتوى على عنصر واحد أبوي.

### كتابة ملفات XML واستخدام التصنيف XmlTextWriter

يمكنك استخدام التصنيف **XmlTextWriter** في كتابة ملفات XML كما يمكنك استخدام تصنيفات أخرى لنفس الغرض مثل التصنيف **XmlDocument** الذي ستعرض له اليوم وتعرضنا له من قبل عند الحديث عن العمل مع الملفات، إلا أنه من المفضل استخدام التصنيف الأول للحصول على أفضل النتائج وأسرعها. يوضح جدول ٩-١ بعض الوظائف والخصائص الموجودة داخل التصنيف **XmlTextWriter** واستخدام كل منها.

جدول ٩-١ خصائص ووظائف التصنيف **XmlTextWriter**

الاسم	الاستخدام
<b>WriteStartDicument</b>	تستخدم هذه الوظيفة في كتابة سطر بداية ملف XML <?xml version="1.0" ?>
<b>WriteStartElement</b>	تستخدم هذه الوظيفة في كتابة رمز البدء لأحد العناصر
<b>WriteEndElement</b>	تستخدم هذه الوظيفة في كتابة رمز انتهاء أحد العناصر
<b>WriteAttributeString</b>	تستخدم هذه الوظيفة في كتابة صفة لأحد العناصر
<b>WriteString</b>	تستخدم هذه الوظيفة في كتابة بيانات نصية داخل أحد العناصر

الاسم	الاستخدام
WriteElementString	تستخدم هذه الوظيفة في كتابة أحد العناصر وبياناته النصية
WriteComment	تستخدم هذه الوظيفة لكتابة تعليق
WriteCData	تستخدم هذه الوظيفة لكتابة الجزء CDATA
Flush	تستخدم هذه الوظيفة لتفريغ محتويات كائن التصنيف XmlTextWriter
Close	إغلاق التصنيف XmlTextWriter حيث يجب إغلاق التصنيف بعد الانتهاء من استخدامه
Formatting	تستخدم هذه الخاصية لتعيين تنسيق الملف وتحتوى على القيمة Formatting.Indented أو Formatting.Non والأخيرة هي القيمة الافتراضية لهذه الخاصية
Indentation	تعيين مسافة الهامش الذى يسبق العناصر الفرعية عن العناصر الأبوية

سنقوم فيما يلى بالتعرف على كيفية استخدام التصنيف XmlTextWriter لإنشاء صفحة ASP.NET تحتوى على كود XML. تابع معنا الخطوات الآتية:

١. قم بإنشاء تطبيق ASP.NET جديد باسم مناسب وليكن WriteXML.
٢. أعد تسمية النموذج Default.aspx إلى اسم مناسب وليكن WriteXML.aspx.
٣. قم بإضافة زر أمر من مربع الأدوات إلى النموذج وأعد تسميته إلى btnGen وقم بتغيير عنوانه إلى Generate XML.
٤. قم بإضافة أداة عنوان أسفل الزر مع تغيير اسمها إلى lbl وحذف محتوياتها.

٥. انقر الزر Source بالجزء السفلي من النموذج ثم قم بتعديل كود الصفحة كما يلي:

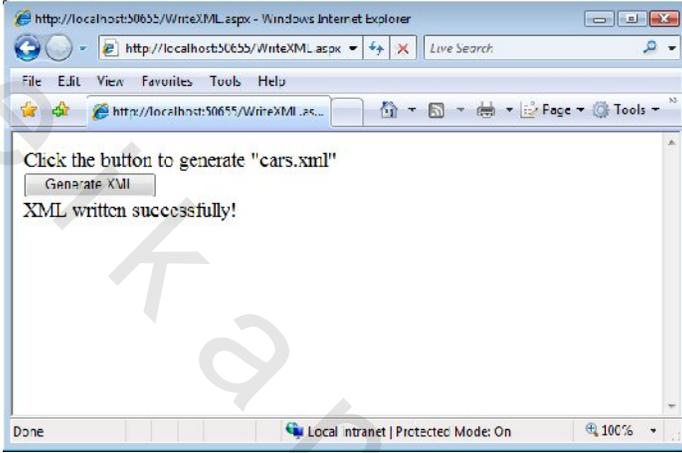
1. <%@ Import Namespace="System.Xml" %>
2. <%@ Page Language="VB" Debug="true" %>
3. <html>
4. <script runat="server">
5. Sub OnWriteClick(Sender As Object, e As EventArgs)
6. Dim xmlWriter as new
7. XmlTextWriter(Server.MapPath("cars.xml"), Nothing)
8. xmlWriter.Formatting = Formatting.Indented
9. xmlWriter.WriteStartDocument()
  
10. xmlWriter.WriteStartElement("cars")
11. xmlWriter.WriteStartElement("car")
  
12. xmlWriter.WriteStartElement("make")
13. xmlWriter.WriteAttributeString("foreign", "true")
14. xmlWriter.WriteString("Honda")
15. xmlWriter.WriteEndElement()
  
16. xmlWriter.WriteElementString("model", "Accord")
17. xmlWriter.WriteElementString("color", "Black")
18. xmlWriter.WriteElementString("year", "1998")
  
19. xmlWriter.WriteEndElement()
20. xmlWriter.WriteEndElement()
  
21. xmlWriter.Flush()
22. xmlWriter.Close()
  
23. lbl.Text = "XML written successfully!"
24. End Sub
25. </script>
26. </body>

27. `<form runat="server">`
28. Click the button to generate "cars.xml"
29. `<br />`
30. `<asp:Button runat="server" id="btnGen" Text="Generate XML" OnClick="OnWriteClick" />`
31. `<br />`
32. `<asp:Label id="lbl" runat="server" />`
33. `</form>`
34. `</body>`
35. `</html>`

وعن هذا الكود، نوضح ما يلي:

- يقع الجزء الهام من الكود داخل إجراء معالجة الحدث `OnWriteClick` في السطور من ٥ إلى ٢٤. ففي السطر رقم ٦ تم تعريف حالة جديدة من التصنيف `XmlTextWriter` وهو الكائن `xmlWriter` والذي يحتوى على معاملين، الأول هو اسم ملف XML الناتج بينما يستخدم الثاني لتعيين مخطط تشفير آخر، وقد استخدمنا هنا `Nothing` لاستخدام التشفير الافتراضى.
- في السطر رقم ٩ تم استدعاء الوظيفة `WriteStartDocument()` لكتابة سطر بداية ملف XML.
- في السطر رقم ١٠ يتم استدعاء الوظيفة `WriteStartElement()` لكتابة بداية العنصر الأبوى `Cars` وفي السطور التالية يتم كتابة بداية العناصر الوليدة.
- في السطر رقم ١٣ يتم استدعاء الوظيفة `WriteAttributeString()` لكتابة الصفة `foreign` وقيمتها داخل العنصر `make`.
- في السطر رقم ١٤ يتم استدعاء الوظيفة `WriteString()` لكتابة بيانات العنصر.
- في السطر رقم ١٥ يتم استدعاء الوظيفة `WriteEndElement()` لكتابة رمز إغلاق العنصر `make`.
- في السطور من ١٦ إلى ١٨ يتم استدعاء الوظيفة `WriteElementString()` لكتابة العنصر وبياناته لأن العنصر في هذه الحالة لا يحتوى على صفات.

- في السطر رقم ٢١ يتم استدعاء الوظيفة **Flush** لحذف محتويات الكائن **xmlWriter** وفي السطر التالي يتم استدعاء الوظيفة **Close** لإغلاق الكائن. والآن يمكنك الحصول على ملف **Cars.xml** الناتج كما يلي:
- ١. قم بتشغيل التطبيق، تحصل على نموذج الويب الموضح في شكل ٩-٢ التالي.



شكل ٩-٢ نموذج الويب لإنشاء ملف XML

٢. انقر زر **Generate XML**، يتم إنشاء الملف **Cars.xml** داخل مجلد التطبيق، ويتحول نص أداة العنوان الموجودة أسفل زر الأمر إلى **XML Written Successfully**.
٣. انتقل إلى المجلد **WriteXML** الموجود على الخادم، تلاحظ وجود الملف **CarsXML**.
٤. قم بفتح الملف داخل المستعرض، تلاحظ احتوائه على نفس الكود السابق (راجع شكل ٩-١ حيث يظهر العنصر **Car** الأول فقط).

### قراءة ملفات XML باستخدام التصنيف XmlTextReader

يمكنك قراءة بيانات XML بما في ذلك الملف الذي قمنا بإنشائه في البند السابق من خلال التصنيف **XmlTextReader** الذي يتيح لك قراءة بيانات الملف فقط. فإذا أردت تعديل

بيانات الملف وكتابته مرةً أخرى، يمكنك استخدام التصنيف `XmIDocument` كما سنرى فيما بعد. إلا أنه من الأفضل استخدام التصنيف الأول إذا أردت القراءة فقط من الملف وذلك لسرعته العالية مقارنةً بالتصنيف `XmIDocument`.

وتتمثل الطريقة المعتادة في استخدام التصنيف `XmITextReader` في إنشاء حالة من هذا التصنيف ثم استدعاء الوظيفة `Read()` باستمرار لقراءة محتويات الملف. وبعد كل عملية استدعاء، يتم ملء كائن التصنيف بالعقدة `Node` التالية داخل الملف. وكل عقدة عبارة عن جزء من عنصر `XML` مثل رمز البداية أو رمز النهاية أو سطر الرأس أو حتى البيانات النصية، حيث يستخدم مصطلح "عقدة" `Node` داخل تصنيفات `.NET`. وذلك لمعاملة ملفات `XML` في معاملة الشجرة `Tree` عند قراءة محتوياتها. وبعد قراءة كل عقدة على حده، يمكنك اختبار محتوياتها من خلال خصائص التصنيف `XmITextReader` المتعددة.

يوضح جدول ٩-٢ التالي أهم الوظائف والخصائص المصاحبة للتصنيف `XmITextReader`.

جدول ٩-٢ وظائف وخصائص التصنيف `XmITextReader`

الاسم	الاستخدام
Name	توضح هذه الخاصية اسم العقدة الحالية وتحتوي على قيمة إذا كان نوع العقدة <code>Element</code> أو <code>Attribute</code>
Value	تعبر هذه الخاصية عن قيمة العقدة الحالية ولا تكون خالية إذا كانت العقدة الحالية من النوع <code>Text</code> أو <code>Attribute</code> أو <code>CDATA</code>
Depth	تعبر هذه الخاصية العمق الحالي داخل شجرة <code>XML</code>
Item	تعبر هذه الخاصية عن قيمة إحدى الصفات
NodeType	تعبر هذه الخاصية عن نوع العقدة الحالية مثل <code>NodeType.Element</code> التي تدل على أن العقدة الحالية عبارة عن عنصر <code>Element</code>

الاسم	الاستخدام
EOF	تقوم هذه الخاصية بإرجاع القيمة True إذا كانت العقدة الحالية في نهاية الملف
Read	تستخدم هذه الوظيفة في قراءة عنصر من ملف XML وتقوم بإرجاع القيمة False إذا وصلت إلى نهاية الملف
MoveToAttribute	تقوم هذه الوظيفة بالانتقال إلى رقم الصفة المحدد
MoveToElement	تقوم هذه الوظيفة بالانتقال إلى العنصر الذي يحتوي على الصفة الحالية
Close	تقوم هذه الوظيفة بإغلاق التصنيف XmlTextReader حيث يجب إغلاق التصنيف بعد الانتهاء من استخدامه

والآن لنتعرف على كيفية استخدام التصنيف XmlTextReader في قراءة محتويات الملف Cars.xml الذي أنشأناه في المثال السابق. تابع معنا الخطوات الآتية:

١. قم بإضافة نموذج جديد إلى التطبيق الحالي مع تغيير اسمه إلى ShowXML.aspx وجعله نموذج بدء التطبيق.

٢. انقر زر Source ثم قم بإدخال الكود التالي:

1. <%@ Page Language="vb" Debug="True" %>
2. <%@ Import Namespace="System.Xml" %>
3. <script runat="server">
4. Sub Page\_Load(Sender As Object, e As EventArgs)
5. Dim xmlReader as new  
XmlTextReader(Server.MapPath("cars.xml"))
6. DisplayXML(xmlReader)
7. xmlReader.Close()
8. End Sub
  
9. Sub DisplayXML(xmlReader as XmlTextReader)
10. Dim attIdx As Integer
11. while (xmlReader.Read())

```
12.         if xmlReader.NodeType = XmlNodeType.Element then
13.             Response.Write("<br>")
14.             Response.Write("<b>Element: </b>" +
15.                             xmlReader.Name)
16.         if xmlReader.HasAttributes then
17.             Response.Write(" <b>Attributes: </b>")
18.             for attIdx=0 to xmlReader.AttributeCount - 1
19.                 ' the Item property retrieves the value for
20.                 each node's attribute
21.                 Response.Write(xmlReader(attIdx))
22.             if attIdx < xmlReader.AttributeCount - 1 then
23.                 Response.Write(", ")
24.             End If
25.         Next
26.     End If
27.     else if xmlReader.NodeType = XmlNodeType.Text then
28.         Response.Write(" <b>Value: </b>" +
29.                         xmlReader.Value)
30.     End If
31. End While
32. End Sub
33. </script>
```

وعن هذا الكود، نوضح ما يلي:

- قمنا بدايةً بإنشاء كائن `XmlTextReader` داخل إجراء معالجة الحدث `Load` بتمرير اسم الملف إلى دالة إنشاء الكائن (السطر رقم ٥) ثم تمرير هذا الكائن إلى الوظيفة `DisplayXML`.
- في السطور من ٩ إلى ٣٢ يتم تعيين كود الوظيفة `DisplayXML` وذلك باستدعاء الوظيفة `Read`، وفي كل مرة يتم قراءة عقدة داخل ملف `XML`.
- في السطر رقم ١٢ يتم اختبار العقد ذات النوع `Element` مثل العقدة `<color>` أو النوع `Text`. كما يمكن البحث عن أنواع أخرى كالنوع `EndElement` مثل `</color>`. يوضح جدول ٩-٣ التالي الأنواع المختلفة من العقد التي يدعمها التصنيف `XmlTextReader`.

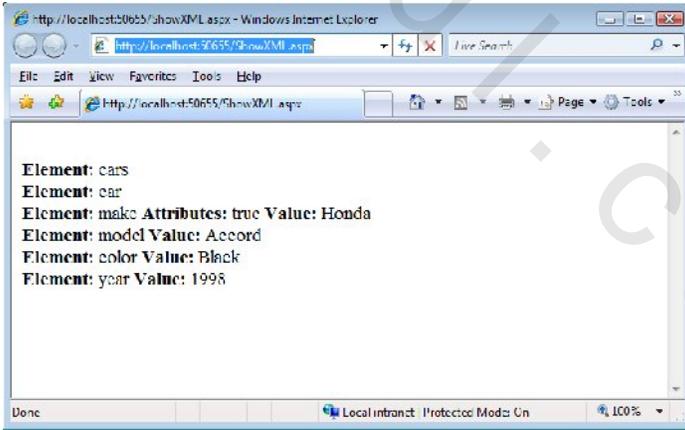
• نجد في منتصف الكود العبارة `Response.Write(xmlReaser[attldix])`

التي يتم من خلالها كتابة بيانات XML.

جدول ٩-٣ أنواع العقد التي يدعمها `XmlTextReader`

النوع	الاستخدام
ELEMENT	يعبر عن رمز البداية بعنصر XML مثل <code>&lt;color primary="true" visible="true"&gt;</code>
ENDELEMENT	يعبر عن رمز نهاية عنصر XML مثل <code>&lt;/color&gt;</code>
TEXT	يعبر عن بيانات نصية داخل عقدة XML
CDATA	يعبر عن الجزء <code>CDATA</code> الخاص بعقدة XML
WhiteSpace	يعبر عن المسافة بين العقد
xmlDeclaration	يعبر عن رأس ملف XML <code>&lt;?xml version="1.0" ?&gt;</code>

والآن قم بتشغيل التطبيق، تحصل على بيانات الملف `Cars.xml` داخل نافذة المستعرض (انظر شكل ٩-٣).



شكل ٩-٣ استخدام التصنيف `XmlTextReader` لقراءة محتويات ملف XML

يمكنك استخدام التصنيف XmlTextReader لقراءة محتويات ملف XML موجود على حاسب آخر على الشبكة المحلية أو شبكة الإنترنت. فقط قم بتمرير عنوان هذا الملف إلى دالة إنشاء كائن التصنيف XmlTextReader.

